Šolski center Celje

Gimnazija Lava

SUDOKU

raziskovalna naloga

Avtor:
Robert PAJEK 4.f

Mentor:
Borut SLEMENŠEK, univ. dipl. inž.

Mestna občina Celje, Mladi za Celje
Celje, 2010

1. Kazalo

2. Kazalo slik

3. Zahvala

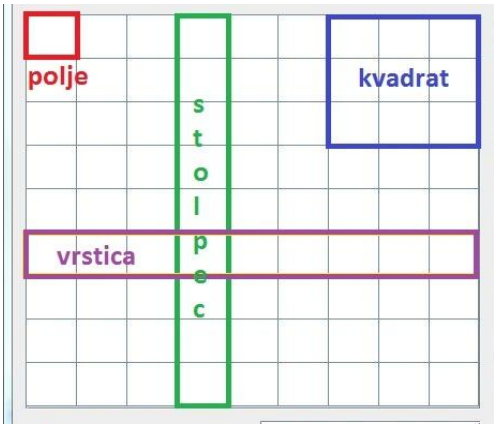
Zahvalil bi se profesorju računalništva in hkrati mentorju te projektne naloge Borutu Slemenšku, za pridobljeno znanje pri njegovih urah in pomoč pri programiranju. Zahvalil pa bi se tudi Tomislavu Viherju za pomoč pri prijavi in obrazložitvi sestavnih delov projektne naloge.

4. Povzetek

V raziskovalni nalogi je opisano kako rešiti sudoku z zaporedjem matematičnih in logičnih metod. Uporabil sem znanje, ki sem ga pridobil v šoli pri predmetu računalništva in z nekaj dodatne literature napisal program, ki reši sudoku. Spodaj sem opisal tudi metode s katerimi pridemo do rešitve. Delno sem izpolnil hipotezo, ki sem si jo zastavil, saj program reši večino sudokujev. Za rešitev težjih sudokujev pa je potrebno uvesti sestopanje (ang. backtracking). Tega mi do sedaj še ni uspelo narediti. Sudoku lahko deluje tudi kot igra, saj ima uporabniku prijazen grafični vmesnik, dodal pa sem tudi nekaj sestavljenih mrež, ki sem jih našel na internetu.

5. Uvod

Moj raziskovalni problem je napisati program, ki bo znal rešiti sudoku.



Sudoku je logična uganka, katere cilj je zapolniti kvadratno mrežo, običajno velikosti 9 x 9, s števili od 1 do 9. Vsako število se lahko pojavi natanko enkrat v vsakem stolpcu, vsaki vrstici in vsakem manjšem kvadratu velikosti 3 x 3. V mreži so nekatera števila že podana. Za rešitev uganke je potreben logičen razmislek in malo potrpežljivosti, pri težjih pa tudi nekoliko kombinatorike, oziroma poskušanja in vračanja.

Slika 1: Obrazložitev mreže

Hipoteza: Sudoku se da rešiti s kombinacijo matematičnih in logičnih metod.

Moja glavna raziskovalna metoda je programiranje v programskem jeziku Java, katera pa je neposredno povezana z metodo zbiranja podatkov.

6. Zbiranje podatkov

Najprej sem prebrskal internet in se pozanimal kaj sploh je sudoku in kako se ga rešuje. Za reševanje sem našel mnogo različnih zanimivih metod. Po premisleku sem se odločil da bom uporabil reševanje z izključevanjem.

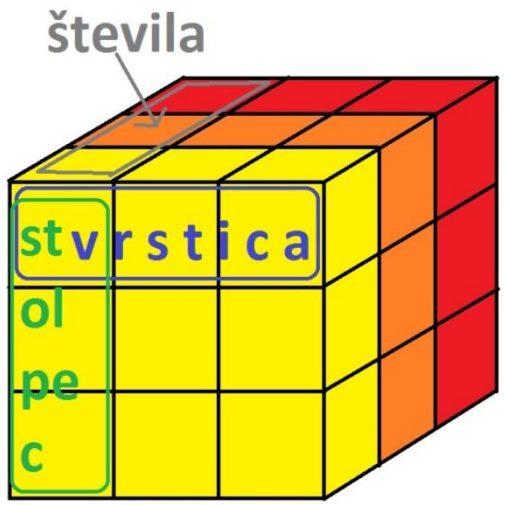
Ta metoda deluje tako da imamo na začetku v vseh poljih vsa števila (od 1 do 9). Ko v polje vnesemo že znano število to naredimo tako da izbrišemo vsa ostala števila. Nato pogledamo vrstico v kateri smo vpisali znano število in iz ostalih polj odstranimo to število (saj se vsako število lahko pojavi samo enkrat v vrstici). Isto naredimo še za kvadrat in stolpec. Tako si pripravimo mrežo števil za nadalnje reševanje. Ko smo odstranjevali vsa ta števila je kar velika verjetnost, da je v katerem polju samo eno število. Zato pregledamo vsa polja in če najdemo polje z enim samim številom ponovimo metodo brisanja vrstic, stolpcov in kvadratov. Ko s ponavljanjem te metode ne pridemo do novih rezultatov, pogledamo če se kje v vrstici pojavi določeno število samo v enem kvadratu. Če najdemo tako število odstranimo to število iz ostalih vrstic v kvadratu. Isto naredimo še za števila v kvadratu stolpca.

7. Programiranje

1. Programski jezik in okolje

Ker se v šoli učimo programski jezik Java sem se odločil da bom programiral v Javi. Pomagal pasi bom z NetBeans IDE 6.7.1 okoljem ki omogoča grafično oblikovanje, vgrajen pa ima tudi prevajalnik, tolmač (interpreter), razhroščevalnik (debuger) in podobna orodja, ki nam pomagajo pri programiranju.

2. Izbira tabele



Slika 2: ponazoritev na tabeli 3x3x3

Na prvi problem sem naletel že takoj na začetku, ko sem razmišljal kam shranjevati števila. Števila so celoštevilske vrednosti (v Javi integer) in te ponavadi predstavimo z tabelo integerjev. To bi bila tabela z 81 polji v vsakem pa bi bilo število 123456789. Ko sem razmišljal kako bom brisal števila sem ugotovil da to ni dobra ideja, saj če želimo izbrisati 6, moramo 6 pomnožiti z 1000 in nato rezultat odšteti od števila (da dobimo 123450789). Nato sem razmišljal da bi števila predstavil z nizi (v Javi String), tako bi lahko enostavno zamenjal »črko« 6 z 0. Ko sem začel pisati program sem ugotovil da je kode zelo veliko saj moraš gledati vsak znak posebej. Na koncu sem prišel do ideje, da bom uporabil tabelo boolean (v boolean lahko zapišemo samo vrednost true ali false(da oz. ne)). Ideja je bila dobra saj je takšne vrednosti zelo preprosto preverjati, vendar se je pojavila težava v strukturi tabele, ki sem jo rešil z tretjo dimenzijo ki predstavlja vrednosti števil. Glej sliko 2. Barve predstavljajo števila, vrstica eno vrstico, stolpec pa en stolpec. Od naše tabele se razlikuje po tem, da imamo mi 9 stolpev, vrstic in števil, tukaj pa sem predstavil samo 3.

3. Metode za reševaje sudokuja

Predstavil bom glavne metode, s pomočjo katerih rešimo sudoku.

Opomba: Zaradi lažje predstave, v nadaljevanju ne bom govoril o spreminjanju vrednosti atributov tretje dimenzije tabele na vrednost false, ampak o brisanju števil iz polja.

```
private void ugotoviEdinoStevilo(){...}
private void brisanjeStevil(){...}
private void edinoVVrstici(){...}
private void edinoVStolpcu(){...}
private void edinoVKvadrantu(){...}
private void triVVrstici(){...}
private void triVStolpcu(){...}
private void brisiDveVrstici(){...}
private void brisiDvaStolpca(){...}
private void brisanjeK(){...}
private void brisanjeVS(){...}
private void izpis(){...}
```

Slika 3: Metode za reševanje sudokuja

1. Metoda »ugotoviEdinoStevilo( )«

Ta metoda pregleda vsa polja v naši tabeli in ugotavlja, če je v katerem polju samo eno število. Deluje tako da preveri koliko true vrednosti je v vsakem polju (za vsako polje šteje vrednosti true v tretji dimenziji). Če je v polju samo en true smo izpolnili naš cilj in našli edino možno vrednost. To vrednost si zapomnimo in pokličemo naslednjo metodo (»brisanjeStevil()«).

```
private void ugotoviEdinoStevilo(){
    for(int i=0;i<9;i++){
        for(int j=0;j<9;j++){
            int stevc=0;
            for(int k=0;k<9;k++){
                if(tab[i][j][k]==true){
                    stevc++;
                    koordinataX=i;
                    koordinataY=j;
                    stevilo3D=k;
                }
            }
            if(stevc==1)brisanjeStevil();
        }
    }
}
```

Slika 4: Metoda: ugotoviEdinoStevilo()

2. Metoda »brisanjeStevil( )«

Ta metoda je nekakšna povezava do metod »brisanjeVS()« in »brisanjeK()«. Z drugimi besedami ko kličemo metodo »brisanjeStevil()« ta pokliče metodi »brisanjeVS()« in »brisanjeK()«.

```
private void brisanjeStevil(){
    for(int i=0;i<1;i++){
        brisanjeVS();
        brisanjeK();
    }
}
```

Slika 5: Metoda: BrisanjeSrevil()

3. Metoda »brisanjeVS( )«

»brisanjeVS()« predstavlja brisanje vrstice in stolpca. Deluje tako da najprej odstrani število ki smo ga prej ugotovili kot »edino število« iz vseh polj vrstice in stolpca, kjer se nahaja to število. Ker število odstrani tudi iz polja kjer naj bi to število bilo, ga na koncu metode dodamo.

```
private void brisanjeVS(){
    for(int i=0;i<9;i++){
        for(int j=0;j<9;j++){
            if(korak)preveriKonecKoraka();
            if(korakKonc)break;
            tab[koordinataX][j][stevilo3D]=false;
            tab[i][koordinataY][stevilo3D]=false;
            tab[koordinataX][koordinataY][stevilo3D]=true;
        }
        if(korakKonc)break;
    }
}
```

Slika 6: Metoda: brisanjeVS()

4. Metoda »brisanjeK( )«

Ko ugotovimo »edino število« vemo da se to število lahko pojavi samo enkrat v kvadratu. Ta metoda nam izbriše število iz vseh polj kvadrata. Ker število odstrani tudi iz polja kjer naj bi to število vilo, ga na koncu doda.

```
private void brisanjeK(){
    for(int h=0;h<1;h++){
        if(korak)preveriKonecKoraka();
        if(korakKonc)break;
        if(koordinataX<=2){                          //prvi stolpec k
            for(int a=0;a<3;a++){
                if(koordinataY<=2){                  //kvadrant  1 1
                    for(int b=0;b<3;b++){
                        tab[a][b][stevilo3D]=false;
                    }
                }
                if(koordinataY>=3&&koordinataY<=5){ //kvadrant  2 1
                    for(int b=3;b<6;b++){
                        tab[a][b][stevilo3D]=false;
                    }
                }
                if(koordinataY>=6){                  //kvadrant  3 1
                    for(int b=6;b<9;b++){
                        tab[a][b][stevilo3D]=false;
```

Slika 7: Metoda: brisanjeK() (izsek programa za prvi stolpec kvadratov)

5. Metoda »edinoVVrstici( )«

S to metodo preglejujemo števila v vsaki vrstici in ugotavljamo če se katero pojavi samo enkrat. To naredimo tako da najprej gledamo vse enke (v našem primeru števila ki imajo v tretji dimenziji v indeksu 0 vrednost true), nato dvojke (števila ki imajo v tretji dimenziji v indeksu 1 vrednost true), trojke itd. in štejemo vse vrednosti true v vrstici. Če je število preštetih vrednosti enako 1 smo našli mesto, kje v vrstici mora biti to število, zato lahko iz polja odstranimo druga števila (ostale vrednosti postavimo na false) in kličemo metodo »ugotoviEdinoStevilo()«.

```
private void edinoVVrstici(){
    int index1=0;                                   //za sistematično preverjanje števil (najprej vse ničle, na
    for(int k=0;k<9;k++){
        for(int i=0;i<9;i++){
            int stevc=0;                            //da vemo katero stevilo gledamo
            for(int j=0;j<9;j++){                   //gremo po vrstici in gledamo koliko istih stevil je
                if(tab[i][j][index1]==true){        //ce je true smo našli stevilo in si zapomnimo koordinate
                    stevc++;                        //štejemo koliko števil je v polju
                    koordinataX=i;
                    koordinataY=j;
                    stevilo3D=index1;
                }
            }
            if(stevc==1){                                                       //če mamo samo 1 stevilo
                for(int ii=0;ii<9;ii++)tab[koordinataX][koordinataY][ii]=false; //ostala števila zbrišemo
                if(korak)preveriKonecKoraka();
                if(korakKonc)break;
                tab[koordinataX][koordinataY][stevilo3D]=true;
                ugotoviEdinoStevilo();                          //gremo v metodo,najdemo to edino število i
            }
        }
        index1++;                                               //gremo gledati naslednje število
    }
}
```

Slika 8: Metoda: edinoVVrstici()

6. Metoda »edinoVStolpcu( )«

Deluje podobno kot metoda »edinoVVrstici()«, le da preglejuje števila v stolpcu, in ne v vrstici.

```
private void edinoVStolpcu(){
    int index1=0;                                   //za sistematično preverjanje števil (najprej vse ničle, na
    for(int k=0;k<9;k++){
        for(int i=0;i<9;i++){
            int stevc=0;                            //da vemo katero stevilo gledamo
            for(int j=0;j<9;j++){                   //gremo po vrstici in gledamo koliko istih števil je
                if(tab[j][i][index1]==true){        //ce je true smo našli število in si zapomnimo koordinate
                    stevc++;                        //štejemo koliko števil je v polju
                    koordinataX=j;
                    koordinataY=i;
                    stevilo3D=index1;
                }
            }
            if(stevc==1){                                                       //ce mamo samo 1 stevilo
                for(int ii=0;ii<9;ii++)tab[koordinataX][koordinataY][ii]=false;     //ostala števila zbrišemo
                if(korak)preveriKonecKoraka();
                if(korakKonc)break;
                tab[koordinataX][koordinataY][stevilo3D]=true;
                ugotoviEdinoStevilo();                          //gremo v metodo,najdemo to edino stevilo i
            }
        }
        index1++;                                               //gremo gledati naslednje število
    }
}
```

Slika 9: Metoda: edinoVStolpcu()

7. Metoda »edinoVKvadrantu( )«

Poišče edino število v kvadratu, deluje pa na podoben način kot prejšni dve metodi. Za razliko od njiju ta metoda preglejuje vsak kvadrant posebej in išče število ki se pojavi samo enkrat.

```
private void edinoVKvadrantu(){
    int stev=11;
    for(int st=0;st<9;st++){                                    //se sprehajamo po kvadrantih in štejemo števila podobno kot
        int jj=6;
        int ii=0;
        for(int l=0;l<3;l++){
            jj-=9;
            ii+=3;
            for(int i=0;i<3;i++){
                jj+=3;
                ii-=3;
                int stevc=0;
                for(int j=0;j<3;j++){
                    for(int k=0;k<3;k++){
                        if (tab[ii][jj][st]==true){
                            stevc++;
                            koordinataX=ii;
                            koordinataY=jj;
                            stev=st;
                        }
                        jj++;
                    }
                    jj-=3;
                    ii++;
                }
                if(stevc==1){                                                   //ce imamo sam 1 število
                    for(int iii=0;iii<9;iii++)tab[koordinataX][koordinataY][iii]=false;     //ostala števila zbrišemo
                    for(int h=0;h<1;h++){
                        if(korak)preveriKonecKoraka();
                        if(korakKonc)break;
                        tab[koordinataX][koordinataY][stev]=true;
                        ugotoviEdinoStevilo();                          //gremo v metodo,najdemo to edino stevilo in brisemo
                    }
```

Slika 10: Metoda: edinoVKvadrantu()

8. Metoda »triVVrstici( )«

Metoda se osredotoči na eno vrstico in jo razdeli na 3 dele (polja 1., 2., 3. kvadranta). Nato išče število ki se pojavi samo v enem od treh delov. Če se tako število pojavi to pomeni da mora biti v tej vrstici, iz tega pa sledi, da ne sme biti v drugih dveh vrsticah istega kvadrata. Zapomnimo si koordinate vrstice, za stolpec pa vzamemo koordinato, ki predstavlja prvi stolpec kvadrata (ker si tako olajšamo delo v nadaljevanju programa; natančneje v metodi »brisiDveVrstici()«). Na koncu metode (če smo našli število ki se pojavi samo v enem delu vrstice) pokličemo metodo »brisiDveVrstici()«.

```
private void triVVrstici(){
    for(int st=0;st<9;st++){
        for(int i=0;i<9;i++){
            int stevc=0;
            int j=0;
            while((j==0)||(j==3)||(j==6)){
                if((tab[i][j][st]==true)||(tab[i][j+1][st]==true)||(tab[i][j+2][st]==true)){
                    stevc++;                        //če najdemo število v treh poljih kvadranta (1,2,3
                    koordinataX=i;
                    koordinataY=j;
                    stevilo3D=st;
                }
                j+=3;                               //da je j prvic 0 potem 3 potem 6 ker imamo v if sta
            }
            if(stevc==1)brisiDveVrstici();          //če so števila v vrstici samo v enem kvadrantu klič
        }
    }

}
```

Slika 11: Metoda: triVVrstici()

9. Metoda »triVStolpcu( )«

Za razliko od metode »triVVrstici« se metoda osredotoči na en stolpec, ga razdeli na 3 dele (polja 1., 2., 3. kvadranta). Nato išče število ki se pojavi samo v enem od treh delov. Če se tako število pojavi to pomeni da mora biti v tem stolpcu, iz tega pa sledi, da ne sme biti v drugih dveh stolpcih istega kvadrata. Zapomnimo si koordinate stolpca, za vrstico pa vzamemo koordinato, ki predstavlja prvo vrstico kvadrata (ker si tako olajšamo delo v nadaljevanju programa; natančneje v metodi »brisiDvaStolpca()«). Na koncu metode (če smo našli število ki se pojavi samo v enem delu stolpca) pokličemo metodo »brisiDvaStolpca()«.

```
    private void triVStolpcu(){
        for(int st=0;st<9;st++){
            for(int i=0;i<9;i++){
                int stevc=0;
                int j=0;
                while((j==0)||(j==3)||(j==6)){
                    if((tab[j][i][st]==true)||(tab[j+1][i][st]==true)||(tab[j+2][i][st]==true)){
                        stevc++;                    //če najdemo število v treh poljih kvadranta (1,2,3
                        koordinataX=j;
                        koordinataY=i;
                        stevilo3D=st;
                    }
                    j+=3;                           //da je j prvic 0 potem 3 potem 6 ker imamo v if sta
                }
                if(stevc==1)brisiDvaStolpca();      //če so števila v stolpcu samo v enem kvadrantu klič
            }
        }

    }
```

Slika 12: Metoda: triVStolpu()

10. Metoda »brisiDveVrstici( )«

V tej metodi odstranimo število iz dveh vrstic kvadrata, kjer to število ne more biti (podatke dobimo iz metode »triVVrstici«).

```
private void brisiDveVrstici(){

    if(koordinataX%3==0){           //ce smo v 1. vrstici kvadranta (brisemo 2. in 3. )
        for(int i=1;i<3;i++){
            for(int j=0;j<3;j++){
                tab[koordinataX+i][koordinataY+j][stevilo3D]=false;
            }
        }
    }
    if(koordinataX%3==1){           //ce smo v 2. vrstici kvadranta (brisemo 1. in 3. )
        for(int i=-1;i<2;i++){
            for(int j=0;j<3;j++){
                tab[koordinataX+i][koordinataY+j][stevilo3D]=false;
            }
            i++;
        }
    }
    if(koordinataX%3==2){           //ce smo v 3. vrstici kvadranta (brisemo 1. in 2. )
        for(int i=1;i<3;i++){
            for(int j=0;j<3;j++){
                tab[koordinataX-i][koordinataY+j][stevilo3D]=false;
            }
        }
    }
}
```

Slika 13: Metoda: brisiDveVrstici()

11. Metoda »brisiDvaStolpca( )«

V tej metodi odstranimo število iz dveh stolpcov kvadrata, kjer to število ne more biti (podatke dobimo iz metode »triVStolpcu«).

```
private void brisiDvaStolpca(){
    if(koordinataY%3==0){           //ce smo v 1. stolpcu kvadranta (brisemo 2. in 3. )
        for(int i=1;i<3;i++){
            for(int j=0;j<3;j++){
                tab[koordinataX+j][koordinataY+i][stevilo3D]=false;
            }
        }
    }
    if(koordinataY%3==1){           //ce smo v 2. stolpcu kvadranta (brisemo 1. in 3. )
        for(int i=-1;i<2;i++){
            for(int j=0;j<3;j++){
                tab[koordinataX+j][koordinataY+i][stevilo3D]=false;
            }
            i++;
        }
    }
    if(koordinataY%3==2){           //ce smo v 3. stolpcu kvadranta (brisemo 1. in 2. )
        for(int i=1;i<3;i++){
            for(int j=0;j<3;j++){
                tab[koordinataX+j][koordinataY-i][stevilo3D]=false;
            }
        }
    }
}
```

Slika 14: Metoda: brisiDvaStolpca()

8. Predstavitev rezultatov

Uspelo mi je napisati program, ki učinkovito reši večino sudokujev. S tem sem delno potrdil hipotezo, saj se večino sudokujev, da rešiti z matematočnimi in logičnimi operacijami. Nekaj sudokujev pa je tudi takšnih, da zahtevajo določeno stopno ugibanja. V programiranju to imenijemo sestopanje (ang. backtracking), kar pa pomeni, da poskušamo uganiti število, ter priti do rešitve. Če nam to ne uspe, se vrnemo v prejšnje stanje in poskusimo z novim številom. Meni tega žal še ni uspelo sprogramirati. Izdelal sem tudi uporabniku prijazen grafični vmesnik, ki omogoča 2 načina vnosa kombinacij števil. Zasnovan je v obliki igre, tako da lahko sudoku rešujemo sami, program pa nam pomaga z »namigi« če se nam kje reševanje ustavi.

Slika 15: Grafični vmesnik programa

9. Zaključek

Cilj, ki sem si ga zadal na začetku sem uspešno dosegel, saj sem napisal program, ki reši sudoku. Imel sem kar nekaj težav; na začetku z izbiro tabele, nato z samimi metodami, pa z hitrim delovanjem le teh. Vse te težave sem uspel rešiti, dodal sem še grafični vmesnik in tako program naredil prijaznejši uporabniku.

Za nadalnje raziskovanje obravnavanega problema bi predlagal dopolnitev programa z metodami, ki bi omogočale sestopanje (ang. backtracking) in tako bi program lahko rešil čisto vsak sudoku.

10. Viri in literatura

Sudoku – Wikipedija, prosta enciklopedija. [online][citirano marec 2010] Dostopno na spletnem naslovu: http://sl.wikipedia.org/wiki/Sudoku

Sudoku Solver by Logic – Solve Methods. [online][citirano marec 2010] Dostopno na spletnem naslovu: http://www.sudokusolver.co.uk/solvemethods.html

SuDoKu | Igrajte SuDoKu – 24ur.com. [online][citirano marec 2010] Dostopno na spletnem naslovu: http://24ur.com/bin/sudoku.php?d=3&section_id=68

Sudoku Puzzles – Solution methods. [online][citirano marec 2010] Dostopno na spletnem naslovu: http://www.puzzle.ro/en/sudoku_solutions.htm

General Java Development Learning Trail – NetBeans Turtoials, Guides and Articles. [online][citirano marec 2010] Dostopno na spletnem naslovu: http://netbeans.org/kb/trails/java-se.html

Java SE Overview – at a Glance. [online][citirano marec 2010] Dostopno na spletnem naslovu: http://java.sun.com/javase/index.jsp

Java Turtorial. [online][citirano marec 2010] Dostopno na spletnem naslovu: http://www.tutorialspoint.com/java/index.htm